Building Data Integration Systems via Mass Collaboration

Robert McCann, AnHai Doan, Vanitha Varadarajan, Alexander Kramnik
Department of Computer Science
University of Illinois, Urbana-Champaign, IL 61801, USA
{rlmccann, anhai, varadara, kramnik}@cs.uiuc.edu

ABSTRACT

Building data integration systems today is largely done by hand, in a very labor-intensive and error-prone process. In this paper we describe a conceptually new solution to this problem: that of *mass collaboration*. The basic idea is to think about a data integration system as having a finite set of parameters whose values must be set. To build such a system the system administrators construct and deploy a system "shell", then ask the users to help the system "automatically converge" to the correct parameter values. This way the enormous burden of system development is lifted from the administrators and spread "thinly" over a multitude of users. We describe our current effort in applying this approach to the problem of schema matching in the context of data integration. We present experiments with both real and synthetic users that show the promise of the approach. Finally we discuss the future work, challenges, and the potential applications of the approach beyond the data integration context.

1. INTRODUCTION

The rapid growth of distributed data on the Web and at enterprises has generated much interest in building data integration systems. Figure 1 shows a data integration system over several sources that list books for sale. Given a user query that is formulated in the query interface (i.e., the *mediated schema*), the system uses a set of *semantic mappings* to translate the query into queries over *source schemas*. Those queries are then executed and the combined results are returned to the user.

Numerous works have been conducted on data integration, both in the database and AI communities (e.g., [6, 11, 7, 8, 3, 2, 9]). Substantial progress has been made in developing conceptual and algorithmic frameworks, query optimization, schema matching, wrapper construction, object matching, and fielding data integration systems on the Web.

Despite this progress, however, today building data integration systems is still largely done by hand in an extremely labor-intensive and error-prone process. The advent of languages and mediums for creating and exchanging semi-structured data, such as XML, OWL, and the Semantic Web, will further accelerate the needs for data integration systems and exacerbate the above problem. Thus it has now become critical to develop techniques that enable the efficient construction and maintenance of data integration systems.

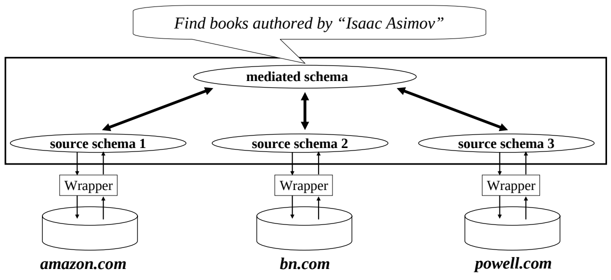


Figure 1: A data integration system in the book domain.

In this paper we describe the MOBS (Mass Collaboration to Build Systems) approach to efficiently building data integration systems. The basic idea underlying our approach is to ask the *users* of a system to *"pay"* for using it by answering relatively simple questions. Those answers are then used to further build the system and expand its capabilities. The following example illustrates our approach:

Example 1: Consider the data integration system in Figure 1. Today, to build such a system we must create the source schemas and the mediated schema and then specify the semantic mappings between these schemas. This is the bare minimum necessary to build a functioning system (with the help of a query processing engine such as the one described in [8]). Notice that any of the above three tasks is well-known to be difficult and time consuming. For example, even with the help of schema matching tools [12] it is still very labor-intensive to manually verify and correct *all* the semantic mappings that the tools suggest.

In the MOBS approach we also start by building the source schemas and the mediated schema. Next we treat the semantic mappings for the mediated-schema attributes as *system parameters*. We assign initial values to these parameters using random assignment or a schema matching tool. We then deploy this system "shell" on the Web and ask users to use it and provide feedback. We use the user feedback to readjust the system parameter values until these values converge.□

In the above example we have focused on treating semantic mappings as system parameters. However, we believe the approach can also be extended to "learn" other system features, such as the wrappers and the source schemas. Note also from the above example that the mass collaboration approach would not replace, but rather *complement* well existing techniques to automate specific tasks in building data integration systems (e.g., schema matching and wrapper construction). In fact we believe it would *amplify* the effects of the current techniques.

Finally, the approach would be applicable to building systems in a broad variety of settings, including enterprise intranets, scientific domains (e.g., bioinformatics), and the Web. For example, within an organization, the employees can collaboratively build and expand a variety of systems that integrate organizational data. Bioinformatists can collaboratively build a data integration system over the hundreds of online bioinformatics sources. Several volunteers in a particular Web domain (e.g., forest preservation) can deploy a system that integrate Wed sources in that domain, by constructing an initial system shell, putting it on the Web, and asking the community to collectively maintain it. This way, the system can be constructed, maintained, and expanded at virtually no cost to any particular entity, but at great benefits for the entire community.

As described, the mass collaboration approach has the potential to dramatically reduce the cost of building data integration systems and spread their deployment in many domains. But the approach also raises numerous challenges. In the rest of the paper we address these challenges. Specifically, we make the following contributions:

- We propose mass collaboration, a conceptually novel approach to the problem of efficiently building data integration systems.

- We describe a solution that applies this approach to schema matching, a critical task that arises while constructing a data integration system.

- We present synthetic and real-world experiments that show the promise of our approach.

- We discuss future work, the challenges, and the potential applications of the approach beyond the context of data integration.

2. THE MOBS APPROACH

Our long-term goal is to use mass collaboration to automate as much as possible *the process* of building a data integration system. As a first step in this direction we study how to automate a major component of that process: finding semantic mappings between the source schemas and the mediated schema.

In this section we describe the MOBS solution to the above schema matching problem. As our running example we will use the simplified BookSeller data integration system in Figure 2. The system has a mediated schema with five attributes: *title, author, price, category*, and *year*, and four sources with schemas $S_1 - S_4$ (Figure 2).

Our task is to find the semantic mappings between the mediated schema and the source schemas $S_1 - S_4$. The MOBS solution to this problem consists of four major activities: *initialization, soliciting user feedback, computing user weights*, and *combining user feedback*. We now describe these activities in detail.

2.1 Initialization

We begin by building a *correct but partial* data integration system. We manually specify the correct mappings from *title* to each of the source schemas. Suppose these mappings are $title = a_1, title = b_1, title = c_1$, and $title = d_1$. The mappings allow us to build a system whose query interface consists of a *single* attribute: *title*. Users can immediately query this system to find books based on their titles.

We note that building an initial correct and partial system is crucial because even at the beginning we must have a functioning system. The system will have limited capabilities but it can already answer user queries *correctly.* If we simply initialize *all* mappings randomly we will probably create an initial system which produces *incorrect* query results. Users are unlikely to start using such a system.

Once we have the correct partial system we want to leverage user feedback to build the rest of the system: that is, to find the correct mappings for the remaining four mediated-schema attributes. To do this, we create *system parameters*, each of which maps a remaining mediated-schema attribute to a source-schema attribute. We consider *all* pairing of mediated-schema attributes with source-schema ones, and thus have the following parameters: $author = a_1, author = a_2, \ldots, author = a_5, author = b_1, \ldots, year = d_6$.

The correct value of a parameter such as $author = a_1$ is "yes" if *author* and a_1 are semantically equivalent (e.g., if a_1 is *writer*), and "no" otherwise.

Finally, we randomly set the initial values of the parameters. Section 4 discusses other initialization methods.

2.2 Soliciting User Feedback

We now deploy the above correct, but partial, system on the Web and ask users to begin using it and providing feedback. Our goal is to use the feedback to make the parameters "converge" to their correct values. When this happens we will have found the correct semantic mappings for the rest of the system.

Currently we solicit user feedback as follows. When the user submits a query to the system (e.g., "find all books whose titles contain 'data integration' "), we make the user "jump through a hoop". That is, the user must answer a question on the correct value of a parameter. Only after the user has answered this question do we display the results of his or her query.

Figure 3 shows a sample question that asks the user whether the attribute named *price* of a source (say, attribute d_3 of source S_4) matches attribute *year* of the mediated schema. This question amounts to soliciting the correct value of parameter $year = d_3$.

The user will answer "yes" or "no" (we are currently adding a third option, "not sure"), after examining the *name* of the attribute, several of its *data instances*, and the *context information* (i.e., the surrounding attributes in this case).

The frequency of "hoop jumping" can be adjusted, anywhere from one "hoop" per query to one "hoop" per several queries. In a related paper on this topic [5] we discuss other types of questions to ask the user as well as ways to entice users to answer questions.

2.3 Computing User Weights

In order to detect malicious or ignorant users we compute a *weight* for each user that measures the quality of his or her feedback. This weight is in the range [0,1], with higher

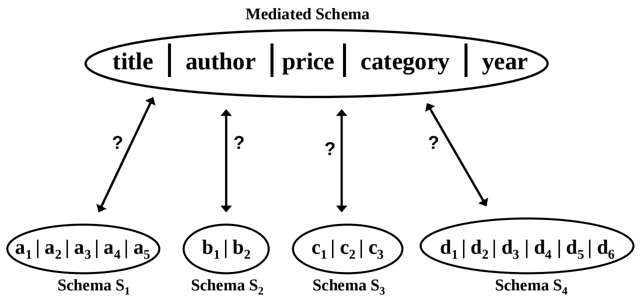
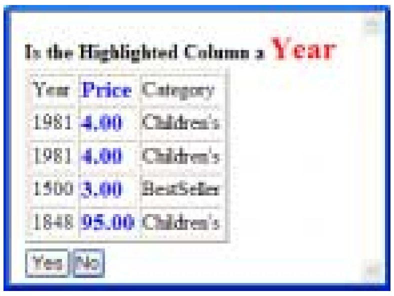

Mediated Schema
title | author | price | category | year
?
?
?
?
a_1 | a_2 | a_3 | a_4 | a_5
Schema S_1
b_1 | b_2
Schema S_2
c_1 | c_2 | c_3
Schema S_3
d_1 | d_2 | d_3 | d_4 | d_5 | d_6
Schema S_4

Figure 2: Using mass collaboration to find the semantic mappings between the attributes of the mediated schema and the source schemas.



Is the Highlighted Column a Year

Year	Price	Category
1981	4.00	Children's
1981	4.00	Children's
1500	3.00	BestSeller
1848	95.00	Children's

Figure 3: A sample question that the system asks the user to answer.

weight indicating higher quality of feedback.

To compute such weights we set aside several sources for *user evaluation.* In the BookSeller example, suppose we set aside the first source, S_1. We create all parameters that are related to this source: $title = a_1, title = a_2, \ldots, title = a_5, author = a_1, \ldots, year = a_5$. We then *manually* assign correct values to these parameters. Next, we solicit user answers on the correct values of these parameters as discussed in the previous subsection. Since we know the correct values of these parameters we can evaluate user answers and compute a user weight.

Consider a user u_1. If the number of answers that u_1 has provided on the parameters coming from the evaluation sources is below a threshold k, then we say that user u_1 has not provided sufficient answers in order to be evaluated, and set $weight(u_1) = 0$. Otherwise we set $weight(u_1)$ to be the faction of u_1's answers that are correct.

To track users we require them to login to use the system. Note that a user needs to login only once; subsequent sessions can be handled automatically by cookies. For any single user we randomly mix the evaluation questions with teaching questions (i.e., questions used to get feedback on unlabeled sources) to ensure that the user does not know which ones are the evaluation questions.

Once a user weight has been computed we stop evaluating the user (we are exploring the option of continual evaluation, see further discussion in Section 3.1). We call a user *trustworthy* if his/her weight is above a threshold w (currently set at 0.65) and *untrustworthy* otherwise. If a user is untrustworthy we do not solicit additional feedback from him/her. In [5] we discuss methods to discourage users from intentionally providing incorrect feedback in an effort to be deemed untrustworthy and avoid answering questions.

2.4 Combining User Feedback

Before describing how to combine user feedback we discuss how we distribute user feedback. Recall that we have set aside source S_1 for evaluation and have manually identified the semantic mappings for this source. We have also identified semantic mappings from the first mediated-schema attribute *title* to all sources. Our job therefore is to solicit user feedback to find semantic mappings from the remaining mediated-schema attributes $author, price, category$, and $year$ to sources $S_2 - S_4$.

We do this sequentially by first finding semantic mappings for *author*, then for *price*, and so on. As soon as we have found the semantic mappings for an attribute, say *author*, we immediately add *author* to the query interface and allow the user to use it to formulate his/her queries. This way user feedback can be shown to have an effect as soon as possible on the query interface, thus enticing users to provide feedback and making the system useful as early as possible.

To find mappings for a mediated-schema attribute, say *author*, we proceed sequentially by finding mappings from *author* to source S_2, then S_3, and then S_4.

Consider source S_2, which has only two attributes b_1 and b_2 (Figure 2). Finding mappings from *author* to S_2 reduces to obtaining the correct values for parameters $author = b_1$ and $author = b_2$. To do this, we solicit user answers in a round-robin fashion, using "hoop jumping" as described in Section 2.2. The first answer goes to $author = b_1$, the second goes to $author = b_2$, the third goes back to $author = b_1$, and so on.

Thus a parameter such as $author = b_1$ will receive a steady stream of "yes" and "no" user answers. Note that these answers are from trustworthy users. We monitor this stream of answers; as soon as a *convergence criterion* is satisfied we assign to parameter $author = b_1$ a value based on the answers it has received so far, and stop soliciting further answers for this parameter.

Suppose parameter $author = b_1$ has received a total of n predictions. The current convergence criterion is to stop if (a) the number of majority answers (either "yes" or "no") reaches $n * 0.65$ and n exceeds 20, or (b) n exceeds 50. In either case we return the majority answer as the value for $author = b_1$. We say parameter $author = b_1$ has *converged* to that value.

We proceed similarly with parameter $author = b_2$. Once all parameters have converged we say the system has *converged* and stop soliciting user feedback.

3. EMPIRICAL EVALUATION

We now describe preliminary experiments that used both synthetic and real user populations to evaluate the MOBS approach.

3.1 Synthetic Experiments

Experimental Setting: We generated a variety of synthetic user populations. A population of 5000 users taken uniformly over [0,1] means that we generated 5000 users and randomly assigned a reliability value from [0,1] to each user. A reliability value of 0.6 means that on average the user answers 6 questions correctly out of 10.

Figure 4.a shows the results over 15 populations (see the figure legend). The populations belong to four classes. *Uniform [0,1]* was described above. *Uniform {0.6,0.4}* means that half the users have reliability value 0.6 and the other

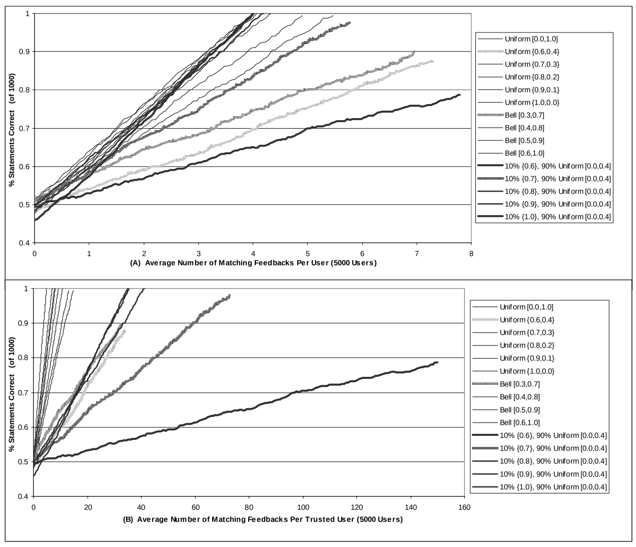


Figure 4: Matching accuracy as a function of average number of user answers, over a broad range of populations.

half 0.4. *Bell [0.3,0.7]* means that the reliability values were generated according to a bell distribution over [0.3,0.7]. Finally, *10% {0.6}, 90% uniform [0.0,0.4]* means 10% of the users have reliability value 0.6 and the rest are assigned values uniformly over [0.0,0.4]. The 15 populations thus represent a broad range of user populations that we expect to commonly occur in practice.

For each population we simulated its interaction with a data integration system that has 10 mediated-schema attributes and 12 sources, with 10 attributes per source. The simulation was carried out using the MOBS mechanism described in Section 2. We set aside 2 sources for evaluation. The system evaluated each user on 10 evaluation questions and deemed that user trustworthy only if he/she answered at least 7 questions correctly.

We solicited user feedback to match the schemas of the remaining 10 sources. Thus we had a total of 10*10*10 = 1000 parameters whose values were to be set by the users. For each parameter we used the stopping criterion described in Section 2. At any timepoint each parameter has an assigned value (either the initial random assignment or the value obtained at convergence time). The *matching accuracy* at that timepoint is then computed as the number of parameters with correct value divided by the total number of parameters.

Matching Accuracy: First we want to know how matching accuracy changes over time and how high it can reach. Figure 4.a plots the *matching accuracy* as the average number of user answers increases, over all 15 populations. Each population is represented with a single line in the figure and the line ends when the system converges.

We use the term *evaluation answers* to refer to those answers that are used to evaluate the reliability of a user, and *teaching answers* to refer to those that are used to set the values of the system parameters. Figure 4.a considers on teaching answers, since the number of evaluation answers is the same for each user (10 in this case).

At the beginning the parameters were initialized randomly, hence matching accuracy around 50% (Figure 4.a). As users provide answers, the accuracy increases linearly and quickly reaches 100% for the first 9 populations in Figure 4.a, with only about 4 teaching answers per user.

Thus, for these populations each user has to provide only 14 answers on average (4 teaching and 10 evaluation) in order to correctly match schemas for the entire system. Note that if a system administrator were to do the schema matching he or she must examine *all* 1000 parameters. Thus, this result supports our argument that we can shift the significant labor (1000 answers) from the administrator to the mass of users, such that each user on average has to do only a small amount of work (14 answers in this case).

For the remaining 6 populations, Figure 4.a shows the matching accuracy in the range 79-99% when all parameters have converged. We note that these populations are quite "unreliable", in the sense that they contain few "good" users and that the highest reliability values of these users are in the range [0.6,0.7]. Since we used only 10 answers for evaluation and used a 0.65 threshold to filter out bad users, we admitted few good users and also admitted some bad users. Thus, for any parameter, when we cut off feedback to it at 50 answers and took the majority vote, there was a significant probability that we ended up with many answers from the bad users, and thus with the wrong parameter value. This explains the less than 100% accuracy at convergence.

Reaching Matching Accuracy of 100%: The above result suggests that if we increase the number of evaluation answers and tighten the convergence criterion (say, to consider cutoff threshold at 100 answers instead of 50), we should be able to increase the matching accuracy. This was indeed the case as we further experimented with these populations. We were able to prove (and confirm experimentally) that if we continuously (a) solicit both evaluation and teaching answers and (b) update user weights and parameter values accordingly, the matching accuracy will converge to 100% provided there are some trustworthy users in the population. The convergence speed obviously depends on the population characteristics. Figure 4.a demonstrates that this speed was quite fast (around 4 teaching answers per user) for a large variety of populations.

Number of Answers per Trusted User: Figure 4.a shows that only a few teaching answers per user (4-8) are required until the system converges to high matching accuracy. However, if the system has decided that a user is not trustworthy it will not solicit answers from that user again. Thus the burden of providing answers will fall mostly to trustworthy users. Hence we want to know how many answers on average such a user must shoulder. Figure 4.b shows that this number of answers remains quite low: around 10 for many populations and under 41 for all populations, except the two outlier populations at 72 and 150. These two populations are the "unreliable" populations discussed above. This result suggests that the burden of feedback for both trustworthy and untrustworthy users remains relatively low.

Effects of Population Size: We have also experimented with population size varying from 50 to 100,000 users, over numerous population topologies. We observed that matching accuracy remains stable across varying sizes. In all cases the time it took to manage the mass collaboration mechanism was negligible. We further observed that, as expected, the number of feedback required per user to reach convergence decreases linearly as the population size increases. This suggests that our approach can scale up to very large populations in all important performance aspects.

3.2 Experiments with Real Users

We have also conducted preliminary experiments with real users and real-world data to evaluate our approach. We built a small data integration system over four real-world book sources. The system is similar to the one shown in Figure 2, with five mediated-schema attributes, four sources, and five attributes per source schema.

We conducted two experiments. In the first one we set aside two sources for testing and built an initial partial system whose query interface had only two attributes: *title* and *author*. We then asked a set of volunteers to use the system and provide feedback. 11 people volunteered, about half of whom had not previously heard about the system.

We simply asked the volunteers to use the system and answer questions to the best of their knowledge. The system deemed seven users as trustworthy. The other four either did not finish evaluation, or provided noisy answers. The system used the trustworthy users' feedback to find semantic mappings for the 3 mediated-schema attributes *price*, *category*, and *year*. It quickly converged and reached 100% matching accuracy. The system was able to expand its query interface to include all five attributes of the mediated schema, thus showing that it can leverage user feedback to expand its capabilities. The average numbers of answers per user and per trustworthy user are 27.45 and 42.71, respectively.

In the second experiment we set aside one source for evaluation and asked for user feedback to match the remaining three sources. We also changed the evaluation and convergence criteria. We asked 5 evaluation questions and admitted users who answered at least 3 correctly. For each parameter we asked for 5 teaching answers and took the majority vote. 8 users volunteered and the system trusted 5. Again the system quickly converged, with the correct values for all parameters except one, thus reaching an accuracy of 26/27 = 96%. The average numbers of answers per user and per trustworthy user are 22.25 and 35.2, respectively.

The real-user experiments therefore provide preliminary evidence that real users can handle the cognitive load of the questions and quickly answer most of them correctly. We are currently designing experiments with much larger data integration settings and many more users. We are also looking for experimental domains that are less well known in order to more thoroughly evaluate the ability of real users to handle relevant questions.

4. DISCUSSION & FUTURE WORK

We have described a basic mass collaboration framework for building data integration systems. We now discuss possible extensions to this framework, as well as additional issues that arise in employing mass collaboration.

Schema Matching: We consider extending our current work on schema matching in several ways. It is clear that the system parameters can be initialized using semi-automatic schema matching tools [12] and that these initial values can be combined with user feedback to achieve faster convergence.

Further, we have focused only on finding one-to-one mappings, such as "location maps to address". We are currently extending our framework to handle more complex mappings, such as "location maps to the concatenation of city and state".

Finally, in the current framework the values of parameters such as $author = b_1$ and $author = b_2$ are obtained *independently* of each other, from the user answers. In many settings, we may know that attribute *author* maps into at most *one* source attribute. Thus, if we already establish that the value of $author = b_1$ is "yes", then we can immediately conclude that $author = b_2$ is "no", without obtaining additional user answers. We are extending our framework to exploit such parameter correlations, in order to minimize the amount of user feedback.

Other Labor-Intensive Tasks: We believe that the current MOBS approach can be extended to handle other data integration issues besides schema matching. Any problem that can be recast as a sequence of "yes"/"no" questions can potentially benefit from this approach. The key will be to break down the problem in such a way that users can handle the cognitive load of answering each individual question. We are currently exploring applying the MOBS approach to the wrapper construction problem.

Mass Collaboration Methodologies: Mass collaboration techniques have been applied to a variety of problems, such as constructing knowledge bases, tech support websites, and word sense disambiguation (see the related work section). However, no systematic study of mass collaboration issues has been conducted. We are currently conducting such a study, which examines the key challenges of mass collaboration in the context of data integration and proposes solutions. Examples of key challenges include how to attract users, how to entice them to give feedback, what types of feedback to solicit, and how to combine their feedback. We provide preliminary discussion of these issues in [5].

Beyond Data Integration: The mass collaboration techniques that we are developing have potential applications beyond just the data integration context. In a sense, the techniques provide a "hammer" that we can use to handle a variety of "nails".

One such "nail" that we are pursuing is marking up data on the Semantic Web. The Semantic Web is advocated as the next-generation World-Wide Web where data is marked up and software programs can exploit the marked-up data to better satisfy information needs of users. Virtually all works on marking up data on the Semantic Web have asked the *owner* of a Web page to mark up the page's data. This approach however often leads to a catch-22 situation: owners are not willing to spend a significant amount of efforts to mark up their pages because they do not see applications that show them the benefits of marking up; on the other hand, applications are not developed because there is no marked-up data.

To break this catch 22, we are exploring a conceptually opposite solution to marking up data: instead of asking the page owner (i.e., the *producer*), we ask the people who visit the page (i.e., the *consumers*) to help mark up the data. Our solution to this problem builds upon the mass collaboration techniques that we are currently developing in the data integration context.

5. RELATED WORK

Our work draws from several related areas, which we discuss below.

Knowledge Base Construction: Our work was inspired by several recent works that attempt to leverage the large volume of Web users to build knowledge bases, tech support websites, and word sense disambiguation ([14, 13], *quiq.com, openmind.org*). The basic idea of these works is to have users contribute facts and rules in some specified language. Our work differs from these in several important aspects. First, in building a knowledge base, potentially *any* fact or rule being contributed constitutes a parameter whose validity must be checked. Thus, the number of parameters can be very high (potentially in the millions) and checking them poses a serious problem. In contrast, the number of (system) parameters in our case is comparatively much smaller and thus potentially much more manageable. Second, such knowledge bases must provide some mechanisms to allow users to immediately leverage the contributed information (to gain some instant gratification effect). Providing such mechanisms in the context of knowledge bases can be quite difficult, because it requires performing inference over a large number of possibly inconsistent or varying-quality facts. Such mechanisms are considerably much simpler in our case, because feedback on the system parameters can immediately affect the query results.

Building Data Integration Systems: The manual construction and maintenance of data integration systems is very labor intensive and error prone. There have been many works on reducing the labor costs of *specific* tasks during the construction process, such as schema matching [12] and wrapper construction (e.g., [10, 1]), but few works on a systematic effort to address cost reduction for the *whole process*, with the exception of [15]. Our work on mass collaboration can be seen as providing a systematic solution to this problem.

Schema Matching: Numerous works have been conducted on schema matching, a fundamental problem in integrating data from heterogeneous sources (see [12] for a survey of recent works). These works employ manually crafted rules and machine learning techniques, with some limited human interaction, to discover semantic mappings. In contrast, our current work leverages the feedback of a multitude of users to find the mappings. To our knowledge, this is the first work on schema matching in this direction.

Autonomic Systems: Our work here is also related to autonomic systems in that data integration systems in the mass collaboration scheme can also exhibit autonomic properties such self-healing and self-improving. The key difference is that autonomic systems have traditionally been thought of as achieving these properties by observing the external environment and adjusting themselves appropriately. In contrast, our systems are observed by the external environments (i.e., the multitude of users) and then are adjusted by them accordingly.

6. CONCLUSION

The current cost of ownership of data integration systems is extremely high due to the need to manually build and maintain such systems. In this paper we have proposed a mass collaboration approach to efficiently build data integration systems. The basic idea is to shift this enormous cost from the producers of the system to the consumers, but spreading it "thinly" over a large number of consumers. We have discussed the challenges of this approach and outlined preliminary solutions. We have also described the current status of our research in this direction and discussed the relationship between this work and several other areas. This research is conducted within the context of the AIDA (Automatically Integrating Data) project at the University of Illinois, whose goal is to build autonomic data integration systems.

7. REFERENCES

[1] N. Ashish and C. Knoblock. Wrapper generation for semi-structured internet sources. *SIGMOD Record*, 26(4):8–15, 1997.

[2] R. Avnur and J. Hellerstein. Continuous query optimization. In *Proc. of SIGMOD '00*, 2000.

[3] J. Chen, D. DeWitt, F. Tian, and Y. Wang. Niagaracq: A scalable continuous query system for internet databases. In *Proc. of SIGMOD '00*, 2000.

[4] A. Doan, P. Domingos, and A. Halevy. Reconciling Schemas of Disparate Data Sources: A Machine Learning Approach. In *Proc. of SIGMOD '01*, 2001.

[5] A. Doan, R. McCann Building Data Integration System: A Mass Collaboration Approach. In *Proc. of the IJCAI-03 Workshop on Information Integration on the Web*, 2003.

[6] H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J. Ullman, and J. Widom. The TSIMMIS project: Integration of heterogeneous information sources. *Journal of Intelligent Inf. Systems*, 8(2), 1997.

[7] L. M. Haas, D. Kossmann, E. L. Wimmers, and J. Yang. Optimizing queries across diverse data sources. In *Proc. of VLDB '97*, 1997.

[8] Z. Ives, D. Florescu, M. Friedman, A. Levy, and D. Weld. An adaptive query execution system for data integration. In *Proc. of SIGMOD '99*, 1999.

[9] C. Knoblock, S. Minton, J. Ambite, N. Ashish, P. Modi, I. Muslea, A. Philpot, S. Tejada. Modeling Web sources for information integration. in *Proc. of the Nat. Conf. on AI (AAAI) '98*, 1998.

[10] N. Kushmerick, D. Weld, and R. Doorenbos. Wrapper Induction for Information Extraction. In *Proc. of the Int. Joint Conf. on AI (IJCAI) '97*, 1997.

[11] A. Y. Levy, A. Rajaraman, and J. Ordille. Querying heterogeneous information sources using source descriptions. In *Proc. of VLDB '96*, 1996.

[12] E. Rahm and P. Bernstein. On matching schemas automatically. *VLDB Journal*, 10(4), 2001.

[13] M. Richardson, R. Aggrawal, and P. Domingos. Building the Semantic Web by mass collaboration. Technical Report UW-TR-03-02-05, Dept. of CSE, Univ. of Washington, 2003.

[14] M. Richardson and P. Domingos. Building large knowledge bases by mass collaboration. Technical Report UW-TR-03-02-04, Dept. of CSE, Univ. of Washington, 2003.

[15] A. Rosenthal, S. Renner, L. Seligman, and F. Manola. Data integration needs an industrial revolution. In *Proc. of the Workshop on Foundations of Data Integration*, 2001.